AMENDMENT TO THE SPECIFICATION:

On Page 1, amend the second paragraph as follows:

Field of the Invention

The present invention relates to a method and system for 3-D (three-dimensional) multiple graphic processing. More particularly, the invention relates to a method and system for improving the parallelization of image processing by Graphic Processing Units (GPUs), based on a unified framework of three parallelization methods, which are time division, image division and object division methods.

On Page 2, amend the last paragraph as follows:

The three-dimensional graphic pipeline architecture breaks-down into segmented stages of CPU, Bus, GPU vertex processing and GPU fragment (pixel) processing. A given pipeline is only as strong as the weakest link of one of the above stages, thus the main bottleneck determines the overall throughput. Enhancing performance is all that is required for reducing or eliminating bottlenecks. The major bottleneck strongly depends on the application. Extreme cases are CAD-like (Computer Aided Design) applications, characterized by an abundance of polygons (vertices), vs. video- game applications having a small polygon count but intensive fragment activity (e.g., texturing). The first class suffers from vertex processing bottlenecks, while the second class suffers from fragment bottlenecks. Both are frequently jammed over the PC bus. Many applications have mixed characteristics, where bottlenecks may randomly alternate between extremes, on a single frame basis.

On Page 3, amend the second and third paragraphs as follows:

In the time division method each GPU renders the next successive frame. It has the disadvantage of having each GPU render an entire frame. Thus, the speed at which each frame is rendered is limited to the rendering rate of a single GPU. While multiple GPUs enable a higher frame rate, a delay can be imparted in the response time (latency) of the system to a user's input. This occurs because, while at any given time, only one GPU is engaged in displaying a rendered frame, each of the GPUs is in the process of rendering one of a series of

frames in a sequence. To maintain the high frame rate, the system delays the user's input until the specific GPU, which first received the signal cycles through the sequence, is again engaged in displaying its rendered frame. In practical applications, this condition serves to limit the number of GPUs that are used in a system. With large data sets, there is another bottleneck, due to the fact that each GPU must be able to access all the data. This requires either maintaining multiple copy operations of large data sets or having possible conflicts in accessing the single copy operation.

~~Image~~ The image division method splits the screen between N GPUs, such that each one displays 1/N of the image. The entire polygon set is transferred to each GPU for processing, however, the pixel processing is significantly reduced to the window size. Image division has no latency issues, but it has a similar bottleneck with large data sets, since each GPU must examine the entire database to determine which graphic elements fall within the portion of the screen allocated to said GPU. ~~Image~~ The image division method suits applications with intensive pixel processing.

On Page 4, amend the first, second and fourth paragraphs as follows:

~~Object~~ The object division method is based on distribution of data subsets between multiple GPUs. The data subsets are rendered in the GPU pipeline, and converted to Frame Buffer (FB) of fragments (sub-image pixels). The multiple FB's sub-images have to be merged (composited) to generate the final image to be displayed. Object division delivers parallel rendering on the level of a single frame of very complex data consisting of a large amount of polygons. The input data is decomposed in the polygon level and re-composed in the pixel level. A proprietary driver intelligently distributes data streams, which are generated by the application, between all GPUs. The rasters, generated by the GPUs, are composited into final raster, and moved to the display. The object division method well suits applications that need to render a vast amount of geometrical data. Typically, these are CAD, Digital Content Creation, and comparable visual simulation applications, considered as~~"viewers,"meaning~~ "viewers", meaning that the data has been pre-designed such that their three-dimensional positions in space are not under the interactive control of the user. However, the user does have interactive control over the viewer's position, the direction of view, and the scale of the graphic data. The user also

may have control over the selection of a subset of the data and the method by which it is rendered. This includes manipulating the effects of image lighting, coloration, transparency and other visual characteristics of the underlying data.

In the above applications, the data tends to be very complex, as it usually consists of massive amount of geometrical entities at the display list or vertex array.

Therefore, there is a need to provide a system which can guarantee the best system performance, while being exposed to high traffic over the PC (Personal Computer) Bus.

On Page 6, amend the first paragraph as follows:

Parallelization is based on an object division mode or on an image division mode or on a time division mode or on any combination thereof. The hardware hub comprises a compositing unit for composing a complete frame from processed portions of the data stream. The hardware hub comprises a hub router for routing polygonal data, for routing a graphic command stream, for routing pixel data and for routing the results of composition, while operating in the object division mode or in the image division mode or in the time division mode or in any combination thereof. The hardware hub comprises a control unit for receiving commands from the Software Hub Driver within the I/O module. The hardware hub comprises a memory unit for storing intermediate processing results of one or more GPUs and data required for composition and transferring the processed data for display.

On Page 8, amend the second paragraph as follows:

The distribution of polygons between multiple GPUs is performed by distributing blocks of data between multiple GPUs and by testing each graphic operation for blocking mode, in which one or more parallelization modes are carried out, thereafter. The data is redirected in a regular non-blocking path to at least one designated GPU, This process is repeated until a blocking operation is detected. Then GPUs are synchronized by performing a flush operation in order to terminate rendering and clean up the internal pipeline in each GPU; performing a composition operation for merging the contents of the Frame Buffers into a single Frame Buffer and by transmitting the single Frame Buffer back to all GPUs. Then the composited complete

frame is terminated at all GPUs, except one or more designated GPUs, whenever a Swap operation is detected and ~~displaying~~ displays the image by means of the one or more designated GPUs. the same data is processed by all GPUs, as long as the blocking mode is active and the Swap operation is not detected. Whenever the blocking mode is inactive, the designated data is further processed by multiple GPUs.